MODELO DE COMPUTADORA

Código de programa

Memoria-Datos (RAM)

En una parte de esa memoria los datos representan el código de un programa (una secuencia de instrucciones) que es ejecutado por el CPU para producir nuevos datos o modificar los que estaban en la memoria

El CPU lee y escribe datos en esa memoria

CPU (procesador)

Esa secuencia de instrucciones es escrita en un lenguaje de programación comprensible y legible por los humanos (Python en nuestro caso) que será llevado a código entendible y ejecutable por el CPU

NOCIÓN INTUITIVA DE PROGRAMA

Describa una secuencia de instrucciones que dadas tres posibles longitudes de segmentos diga si pueden formar un triángulo

- *Lea un dato número entero, guardémoslo en una memoria que le llamenos A*
- *Lea otro dato número entero, guardémoslo en una memoria que le llamenos B*
- *Lea otro dato número entero, guardémoslo en una memoria que le llamenos C*
- *A, B y C pueden formar un triángulo si estas tres expresiones son verdaderas:*
  - *A + B es mayor que C*
  - *B + C es mayor que A*
  - *A + C es mayor que B*

¿QUÉ ES PYTHON?

Es el Lenguaje de Programación que usaremos para escribir nuestros programas.

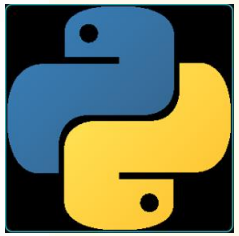
¿Por qué Python?.

Es es uno de los lenguajes más populares y el más utilizado por la comunidad para escribir aplicaciones de Ciencia de Datos y de Inteligencia Artificial

Hay una gran comunidad que desarrolla, mantiene y comparte cantidad de código y bibliotecas de código que incrementan las capacidades de Python. No reiventar el agua tibia.

Python es sencillo para empezar y que puedan luego ir ampliando y aprendiendo y desarrollando las capacidades de programación de modo incremental

Python es divertido



TIPOS DE DATOS BÁSICOS EN PYTHON

Tipos `string`, `int`, `float` y `bool`

Los ladrillos de partida con los que se arma toda aplicación.
Tienen una interpretación , representación y uso predefinidas e integradas en el lenguaje (built-in)

El TIPO CADENA (string)

Secuencias de caracteres entre comillas simples ‘ o dobles “

```
‘Hola Python‘
```

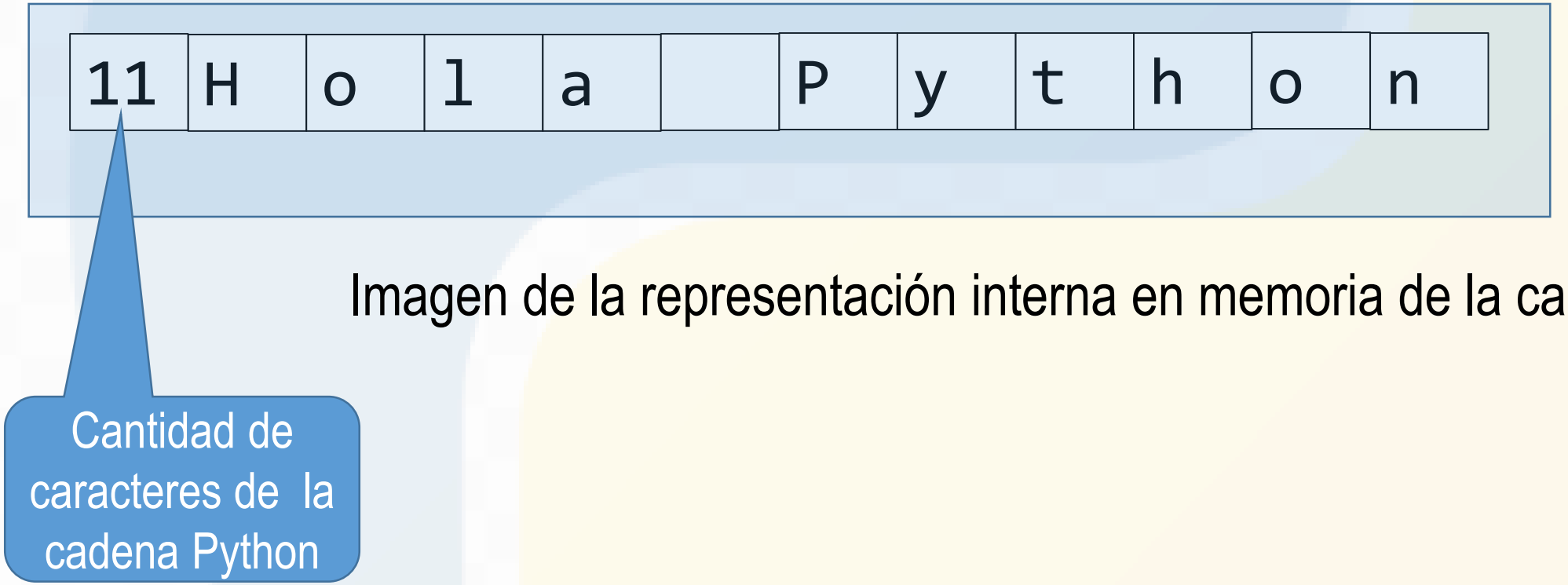


Imagen de la representación interna en memoria de la cadena

DEMO

OPERACIONES CON CADENAS (1)

Operación / Método	Descripción	Ejemplo
`+`	Concatenar dos strings	`"hola" + " mundo"` → `"hola mundo"`
`*`	Repetir el string n veces	`"hola" * 3` → `"holaholahola"`
`in`	Verificar si un substring está contenido	`"a" in "casa"` → `True`
`len()`	Obtener la longitud del string	`len("hola")` → `4`
`str[index]`	Acceder al carácter en la posición index	`"hola"[1]` → `"o"`
`str[start:end]`	Substring desde start hasta end (no inclusivo)	`"hola"[1:3]` → `"ol"`
`lower()`	Convertir todos los caracteres a minúsculas	`"Hola".lower()` → `"hola"`
`upper()`	Convertir todos los caracteres a mayúsculas	`"Hola".upper()` → `"HOLA"`

OPERACIONES CON CADENAS (2)

`capitalize()`	Capitalizar la primera letra	`"hola".capitalize()` → `"Hola"`
`swapcase()`	Cambiar mayúsculas por minúsculas y viceversa	`"HoLa".swapcase()` → `"hOlA"`
`count(sub)`	Contar cuántas veces aparece un substring	`"banana".count("a")` → `3`
`find(sub)`	Encontrar la primera posición de un substring, devuelve -1 si no se encuentra	`"banana".find("na")` → `2`
`replace(old, new)`	Reemplazar substring old por new	`"hola mundo".replace("mundo", "Python")` → `"hola Python"`
`strip()`	Quitar espacios en blanco al inicio y fin	`" hola ".strip()` → `"hola"`
`split(sep)`	Dividir en lista de strings usando sep como separador	`"a,b,c".split(",")` → `["a","b","c"]`
`join(iterable)`	Unir elementos de iterable con el string como separador	`",".join(["a","b","c"])` → `"a,b,c"`

El TIPO ENTERO (int)

Operación / Operador	Descripción	Ejemplo	Resultado
`+`	Suma	`5 + 3`	`8`
`-`	Resta	`5 - 3`	`2`
`*`	Multiplicación	`5 * 3`	`15`
`/`	División (resultado float)	`5 / 2`	`2.5`
`//`	División entera (cociente)	`5 // 2`	`2`
`%`	Módulo (resto de la división)	`5 % 2`	`1`
`**`	Potencia	`2 ** 3`	`8`
`-` (unario)	Negación	`-5`	`-5`

El TIPO FLOAT (float)

Operación / Método	Descripción	Ejemplo	Resultado
`+`	Suma	`3.14 + 2.71`	`5.85`
`-`	Resta	`10.5 - 4.2`	`6.3`
`*`	Multiplicación	`2.5 * 4.0`	`10.0`
`/`	División (resultado float)	`10.0 / 3.0`	`3.3333333333333335`
`//`	División entera (cociente)	`10.5 // 4`	`2.0`
`%`	Módulo (resto de la división)	`10.5 % 4`	`2.5`
`**`	Potencia	`2.0 ** 3.0`	`8.0`
`.is_integer()`	Método que devuelve `True` si el float es un entero (sin decimal)	`(3.0).is_integer()`	`True`
`.as_integer_ratio()`	Devuelve la fracción exacta que representa el float	`(15.23).as_integer_ratio()`	`(1523, 100)`
`.hex()`	Representa el float en notación hexadecimal	`(56.32).hex()`	`'0x1.c7p+5'`
`float()`	Conversión a float desde otros tipos	`float("3.14")`	`3.14`

El TIPO Booleano (bool)

Operación / Operador	Descripción	Ejemplo	Resultado
`and`	Y lógico: True si ambos operandos son True	`True and False`	`False`
`or`	O lógico: True si al menos un operando es True	`True or False`	`True`
`not`	Negación lógica: invierte el valor booleano	`not True`	`False`
`==`	Igualdad: True si operandos son iguales	`5 == 5`	`True`
`!=`	Diferente: True si operandos no son iguales	`5 != 3`	`True`
`<`	Menor que	`3 < 5`	`True`
`>`	Mayor que	`5 > 3`	`True`
`<=`	Menor o igual que	`3 <= 3`	`True`
`>=`	Mayor o igual que	`5 >= 3`	`True`
`bool(x)`	Convierte cualquier valor a booleano	`bool(0)`	`False`
		`bool("texto")`	`True`

Tabla de verdad

AND (`and`):

and	True	False
True	True	False
False	False	False

OR (`or`):

or	True	False
True	True	True
False	True	False

NOT (`not`):

a	not a
True	False
False	True

VARIABLES y ASIGNACION

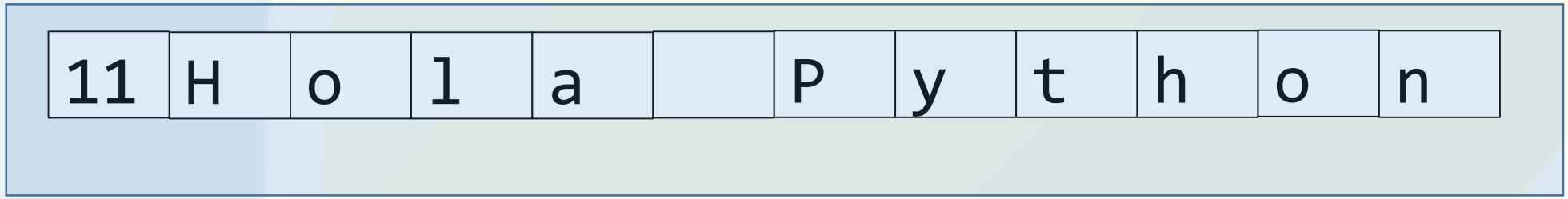
```
a = "Hola Python"
```

Nombre, parte izquierda de la asignación

Expresión que computa un valor, parte derecho de la asignación

Cada variable tiene su espacio en memoria (que Python controla) para guardar el valor (según el tipo) asignado a la variable

a



SINTAXIS PARA LOS NOMBRES DE VARIABLES

- El nombre debe comenzar con una letra (a-z, A-Z) o un guion bajo (_). No puede comenzar con un número ni otros caracteres especiales.
- Después del primer carácter, el nombre puede contener letras, números (0-9) y guiones bajos (_).
- Los nombres son sensibles a mayúsculas y minúsculas, por lo que `variable`, `Variable` y `VARIABLE` son diferentes.
- No se pueden usar palabras reservadas del lenguaje (como `if`, `for`, `def`, `class`, etc.) como nombres de variables o funciones.
- No pueden contener espacios ni caracteres especiales como guiones (-), arrobas (@), etc.

```
cateto1        cateto2      valor_mayor
```